|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**Лабораторная работа № 5**

|  |  |
| --- | --- |
| **Тема Алгоритмы численного интегрирования**  **Студент Якуба Д. В.**  **Группа ИУ7-43Б**  **Оценка (баллы)**  **Преподаватель Градов В. М.** |  |

Москва.

2020 г.

Оглавление

[Лабораторная работа по теме «Алгоритмы численного интегрирования» 2](#_Toc39967811)

[Тема: 2](#_Toc39967812)

[Цель работы: 2](#_Toc39967813)

[Задание: 2](#_Toc39967814)

[Входные данные: 3](#_Toc39967815)

[Выходные данные: 3](#_Toc39967816)

[Описание алгоритма 3](#_Toc39967817)

[Результат 5](#_Toc39967818)

[Алгоритм вычисления n корней полинома Лежандра n-ой степени 5](#_Toc39967819)

[Влияние количества выбираемых узлов сетки по каждому направлению на точность расчётов 6](#_Toc39967820)

[Контрольные вопросы 6](#_Toc39967821)

[1. 6](#_Toc39967822)

[2. 6](#_Toc39967823)

[3. 6](#_Toc39967824)

[4. 6](#_Toc39967825)

[Код программы 6](#_Toc39967826)

Лабораторная работа по теме «Алгоритмы численного интегрирования»

Тема:

Построение и программная реализация алгоритмов численного интегрирования.

Цель работы:

Получение навыков построения алгоритма вычисления двукратного интеграла с использованием квадратурных формул Гаусса и Симпсона.

Задание:

Работа основывается на материалах лекции 5 и 6.

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра .

, где

Входные данные:

Выходные данные:

Значение интеграла, указанного выше, вычисленного методом последовательного интегрирования по одному направлению – с использованием формулы Гаусса, и по-другому – с использованием формулы Симпсона.

График зависимости .

Описание алгоритма

Полагая, что:

Данная система даёт 2n соотношений для определения 2n неизвестных

Система:

Указанная система является нелинейной, и её решения находятся довольно трудно. Воспользуемся полиномами Лежандра:

Узлами формулы Гаусса являются нули данного полинома, а можно найти из заданной выше системы уравнений.

При вычислении интеграла на произвольном интервале , для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной следующим образом:

В таком случае получаем конечную формулу для произвольного интервала интегрирования :

Также, для произвольного интервала формула Симпсона:

Указанные методы можно применять для приближённой оценки двукратного интеграла. Для прямоугольной области будем иметь:

*При этом:*

По каждой из осей введём некоторую сетку узлов. Каждый однократный интеграл будем вычислять по квадратурным формулам. Для разных направлений можем использовать квадратурные формулы разных порядков точности.

Для формулы Гаусса будем иметь:

.

Результат

Алгоритм вычисления n корней полинома Лежандра n-ой степени

Процедура вычисления корней полиному Лежандра произвольной степени выполняется численным методом, например, можно применить метод половинного деления. Сам полином строиться по рекуррентной формуле:

Это рекуррентное соотношение является одним из полезных свойств полинома Лежандра.

Для начала процесса используются полиномы

Процедура повторяется до тех пор, пока не будут найдены все n корней полинома. При этом учитывается свойство полиномов, согласно которому все эти корни располагаются на интервале [-1; 1], и они все действительны и различны, то есть кратные корни отсутствуют.

Влияние количества выбираемых узлов сетки по каждому направлению на точность расчётов

Контрольные вопросы

1.

2.

3.

4.

Код программы

import numpy as np  
from numpy.polynomial.legendre import \*  
from numpy.linalg import eigvalsh  
from math import cos, sin, exp, pi  
  
  
def legendre(deg):  
 c = np.array([0]\*deg + [1])  
 m = legcompanion(c)  
 x = eigvalsh(m)  
 dy = legval(x, c)  
 df = legval(x, legder(c))  
 x -= dy/df  
 fm = legval(x, c[1:])  
 fm /= np.abs(fm).max()  
 df /= np.abs(df).max()  
 w = 1/(fm \* df)  
 w = (w + w[::-1])/2  
 x = (x - x[::-1])/2  
 w \*= 2. / w.sum()  
 return x, w  
  
  
def toResolvation(parameter):  
 subcurFunction = lambda x, y: 2 \* cos(x) / (1 - (sin(x) \*\* 2) \* (cos(y) \*\* 2))  
 curFunction = lambda x, y: (4 / pi) \* (1 - exp(-parameter \* subcurFunction(x, y))) \* cos(x) \* sin(x)  
 return curFunction  
  
  
def Simpson(curFunction, start, end, num):  
 h = (end - start) / (num - 1)  
 x = start  
 output = 0  
 for i in range((num - 1) // 2):  
 output += curFunction(x) + 4 \* curFunction(x + h) + curFunction(x + 2 \* h)  
 x += 2 \* h  
 output \*= h / 3  
 return output  
  
  
def toSingleTemp(twoParSolvation, imput):  
 return lambda x: twoParSolvation(imput, x)  
  
  
def TempToX(parT, start, end):  
 return (end + start) / 2 + (end - start) \* parT / 2  
  
  
def Gauss(curFunction, end, start, num):  
 legArr = legendre(num)  
 output = 0  
 for i in range(num):  
 output += (start - end) / 2 \* legArr[1][i] \* curFunction(TempToX(legArr[0][i], start, end))  
 return output  
  
  
def twoParTag(curFunction, limits, num, integrators):  
 interior = lambda x: integrators[1](toSingleTemp(curFunction, x), limits[1][0], limits[1][1], num[1])  
 return integrators[0](interior, limits[0][0], limits[0][1], num[0])  
  
  
def main():  
 parameter = int(input("Par: "))  
 NSimpson = int(input("n(Simpson): "))  
 MGauss = int(input("m(Gauss): "))  
 output = twoParTag(toResolvation(parameter), ((0, pi / 2), (0, pi / 2)), (NSimpson, MGauss), (Simpson, Gauss))  
 print(output)  
  
  
main()